Modelos Probabilistas Aplicados

Johanna Bolaños Zúñiga

Matricula: 1883900

Tarea 5

1. Introducción

En este trabajo se presentan algunos algoritmos para la generación pseudoaleatoria de números con distribución normal y uniforme. Se utilizan también pruebas de normalidad y uniformidad para verificar que los datos generados siguen este tipo de distribuciones. Se trabaja en el software R versión 4.0.2 [7] y el código empleado se encuentran en el repositorio GitHub [1].

2. Generación de números pseudoaleatorios uniformes

De acuerdo a Hiller [5] los números generados por una computadora no se deben llamar números aleatorios porque son predecibles y se pueden reproducir, dado el generador de números aleatorios que se use. Por ello, en ocasiones se les llama números pseudoaleatorios. Los números pseudoaleatorios son los generados por medio de la computadora (algoritmo), el cual es un proceso que parece producir números al azar, pero no lo hace realmente. Casi cualquier plataforma computacional sabe generar enteros pseudoaleatorios dentro de un rango predeterminado desde cero hasta un máximo (que suele ser una potencia de dos). A partir de ellos, se puede generar otro tipo de valores pseudoaleatorios con aritmética simple [6].

Para la generación de números pseudoaleatorios uniformes (entre $(0, 1)$) a los cuales denotaremos con r_i, independientemente del algoritmo o procedimiento que se utilice para su generación, lo importante son los números que genera, ya que estos deben poseer ciertas características que aseguren su validez [3]. Dichas características son:

- Deben estar uniformemente distribuidos.
- Ser estadísticamente independientes.
- Reproducibles.
- Su periodo o ciclo de vida debe ser largo.
- Deben ser generados a través de un método que no requiera mucha capacidad de almacenamiento de la computadora.
- Se inicia con una semilla X_0 y, a partir de esta, se van generando $X_1, X_2, X_3, \ldots$.

Dada la importancia de contar con un conjunto suficientemente grande de números pseudoaleatorios, existen diferentes algoritmos determinísticos para obtenerlos. En este trabajo nos enfocaremos en el algoritmo generador congruencial mixto, ya que es el más utilizado. Este algoritmo genera una secuencia de números enteros por medio de la siguiente ecuación recursiva:

$$X_{i+1} = (aX_i + c) \bmod m \qquad i = 0, 1, 2, \ldots, N \tag{1}$$

donde a (constante aditiva), c (constante multiplicativa) y m (módulo o longitud del ciclo) son enteros positivos, X es la secuencia de valores pseudoaleatorios y X_0 es la semilla. Como se mencionó anteriormente, los números que se generan son enteros, por lo cual, de acuerdo a García [4], para obtener números pseudoaleatorios uniformes se requiere de la ecuación 2.

$$r_i = \frac{X_i}{m-1} \tag{2}$$

A modo de ejemplo, se consideró generar $n = 8$ números pseudoaleatorios con los siguientes parámetros: $X_0 = 100$, $a = 6$, $c = 22$ y $m = 30$. La secuencia de los números pseudoaleatorios uniformes r_i se muestran en la tabla 1. Para determinar si los datos obtenidos del generador siguen una distribución uniforme, se utilizó la función `uniform.test`. Después de aplicar la prueba, se obtuvo que $p - value = 0{,}945$ es mayor que 0,05 lo que significa que los datos obtenidos al parcer siguen una distribución uniforme.

De la anterior experimentación se pueden realizar las siguientes inferencias: en el cuadro 1, podemos observar que el número generado en X_3 y en X_8 son los mismos y que al aplicar la prueba de uniformidad nos arrojó una advertencia, lo cual podría indicar que, de acuerdo a García [4], si continuamos generando más números con los parámetros dados, estos se repetirían. Para comprobar lo anterior, se realizó otra experimentación donde se generaron 30 números pseudoaleatorios con los mismos parámetros iniciales, y al aplicar la prueba de uniformidad con el comando `uniform.test`, el $p - value = 0{,}002292$ fue menor que 0,05, lo que significa que los datos obtenidos al parecer no siguen una distribución uniforme.

Cuadro 1: Números pseudoaleatorios uniformes del generador $X_{i+1} = (6X_i + 22) \bmod 30$

i	X_i	r_i
1	22	0,7586207
2	4	0,1379310
3	16	0,5517241
4	28	0,9655172
5	10	0,3448276
6	22	0,7586207
7	4	0,1379310
8	16	0,5517241

test.txt

```
        Chi-squared test for given probabilities

data:  hist.output$counts
X-squared = 0.75, df = 4, p-value = 0.945

Warning message:
In chisq.test(x = hist.output$counts, p = probs) :
  Chi-squared approximation may be incorrect
```

test.txt

```
        Chi-squared test for given probabilities

data:  hist.output$counts
X-squared = 24, df = 8, p-value = 0.002292

Warning message:
In chisq.test(x = hist.output$counts, p = probs) :
  Chi-squared approximation may be incorrect
```

De acuerdo a Hiller [5], García [4] y Coss [3], para que el generador de números pseudoaleatorios uniforme sea eficiente, se requiere que los parámetros X_0, a, c y m cumplan ciertas condiciones y sugieren lo siguiente:

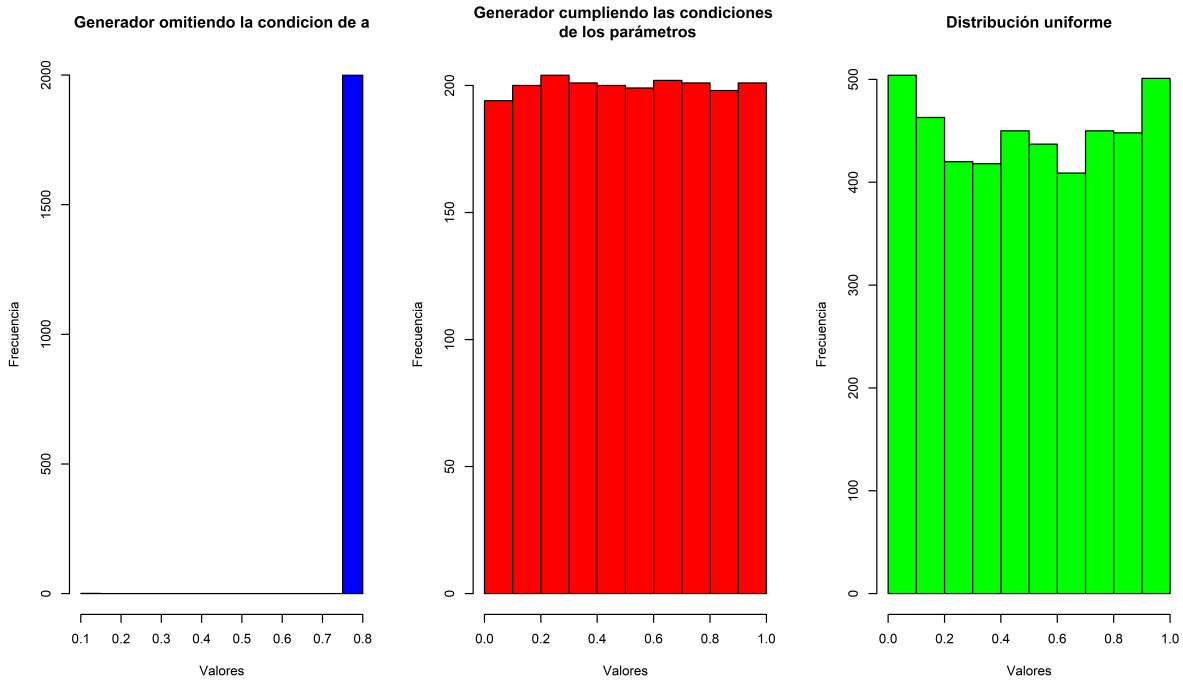

Figura 1: Comparativo de histogramas con el generador omitiendo una condición (azul), cumpliendo todas las condiciones (rojo) y el obtenido por la función `runif` (verde)

- $m = p^d$, donde p es la base del sistema (binario, decimal, hexadecimal, etc.) que se está utilizando y d el número de bits que tiene una palabra de computadora en ese sistema. Para nuestro caso, $m = 2^g$, donde g es entero.
- $a = 1 + 4k$, donde k debe ser entero. Esta expresión hace que a tome valores enteros impares.
- c deber ser un entero impar y relativamente primo[1] a m.
- La semilla X_0 puede tomar cualquier valor, porque sólo afecta a la sucesión en el punto en el que comienza y no en la progresión de los números.

De acuerdo a lo anterior, se realizaron 2 experimentos, uno donde se omite uno de las condiciones y otro donde se cumplen. En ambos casos se consideró generar $n = 2000$ números pseudoaleatorios, para el primer caso, se omite la condición del parámetro a y se contemplaron los siguientes parámetros: $X_0 = 55$, $a = 8,000$, $c = 2,651$ y $m = 2,048$ y, para el segundo caso, los parámetros son: $X_0 = 70$, $a = 8,001$, $c = 2,651$ y $m = 2,048$. En la figura 1, se muestra el comparativo de los histogramas de la experimentación realizada.

Como podemos observar en figura 1, al omitir una de las condiciones en el valor de los parámetros, los números generados no siguen una distribución uniforme, sin embargo, cuando sí se cumplen se puede

[1] Dos números son relativamente primos si su factor común más grande es 1

apreciar que los números generados tienden a seguir esta distribución, por lo tanto, de no cumplirse alguna de las condiciones, el generador no será eficiente. Mediante la prueba `uniform.test`, también se puedo apreciar que los datos del caso donde se omiten una condición, calculó un $p - value = 2{,}2e^{-16}$ y donde se cumplen todas las condiciones el $p - value = 1$.

Otro de los métodos para generar números pseudoaleatorios uniformes es el algoritmo congruencial cuadrático, el cual tiene la siguiente ecuación recursiva:

$$X_{i+1} = (aX_i^2 + bX_i + c) \bmod m \qquad i = 0, 1, 2, \ldots, N \tag{3}$$

De igual forma, se pueden generar los números uniformes mediante la ecuación 2. García [4] menciona también que para que el generador de números pseudoaleatorios uniforme sea eficiente, se requiere que los parámetros a, b, c y m cumplan con las siguientes condiciones:

- $m = 2^g$, donde g es entero.
- a debe ser un número par.
- c debe ser un número impar
- $(b - 1) \bmod 4 = 1$

3. Generación de números pseudoaleatorios distribuidos normalmente

En la sección anterior se mostraron diferentes algoritmos para obtener números pseudoaleatorios uniformemente distribuidos, lo que significa que todos los números tienen la misma probabilidad de aparecer en el resultado, sin embargo, hay casos en los que se hace necesario generar valores aleatorios que sigan otros tipos diferentes de distribución, como por ejemplo la distribución normal (o gaussiana).

Para la generación de los números pseudoaleatorios con distribución normal utilizaremos el método de transformación de Box-Muller [2], ya que a partir de números aleatorios uniformemente distribuidos, genera pares de números aleatorios independientes con distribución normal estándar. En el algoritmo 1 se describe el procedimiento para la generación de este par de números, donde U_1 y U_2 son variables aleatorias independientes que están uniformemente distribuidas en el intervalo $(0, 1]$, z_0 y z_1 son variables aleatorias independentes con una distribución normal con desviación típica 1.

A manera de ejemplo, se realizó una experimentación donde se generaron $n = 4,500$ replicas con los números pseudoaleatorios distribuidos normalmente, con media ($\mu = 20$), desviación estándar ($\sigma = 15$).

Algoritmo 1: Algoritmo generador de pares de números pseudoaleatorios distribuidos normalmente (*tranformación de Box-Muller*)

Entrada: media (μ), desviación estándar (σ);

Salida: Dos números pseudoaleatorios distribuidos normalmente (z_0, z_1);

$U \leftarrow$ `runif(2)`;

$z_0 \leftarrow \sqrt{-2 \ln U_1} * \cos(2\pi U_2)$;

$z_1 \leftarrow \sqrt{-2 \ln U_1} * \sin(2\pi U_2)$;

`datos` $\leftarrow [z_0, z_1]$;

return $datos * \sigma + \mu$;

Los resultados obtenidos, se muestran en la figura 2, en la cual podemos observar que los histogramas de distribuciones parecen semejantes, lo que podría significar que el algoritmo utilizado al parecer está generando números pseudoaleatorios distribuidos normalmente.

Para determinar si las n replicas de los números generados siguen una distribución normal, se aplica la prueba de normalidad *Shapiro–Wilks* con la función `shapiro.test`. Esta prueba plantea la hipótesis nula que una muestra proviene de una distribución normal y una hipótesis alternativa que sostiene que la distribución no es normal. Para aceptar la hipótesis nula el $p - value$ debe ser mayor a 0,05. De acuerdo al resultado obtenido de la prueba, $p - value = 0{,}7955$, lo que significa que se aprueba la hipótesis nula y, por lo tanto, los datos obtenidos siguen una distribución normal.

`test.txt`

```
        Shapiro-Wilk normality test

data:  pseudos
W = 0.9997, p-value = 0.7955
```

3.1. Diferencias entre las variables z_0 y z_1 y comportamiento entre U_1 y U_2

Se llevaron a cabo diversos experimentos para determinar el comportamiento de los datos obtenidos por el generador propuesto, en los cuales se consideraron 3 escenarios diferentes. En el escenario 1, las variables aleatorias uniformes U_1 y U_2 no son independientes; en el escenario 2 solo se considera la variable z_0 y, en el escenario 3 solo se considera la variable z_1. Se continua con los parámetros $n = 4,500$, $\mu = 20$, $\sigma = 15$.

Para el escenario 1, se consideran dos casos, dependencia directa ($U_2 = 2U_1$) y dependencia in-

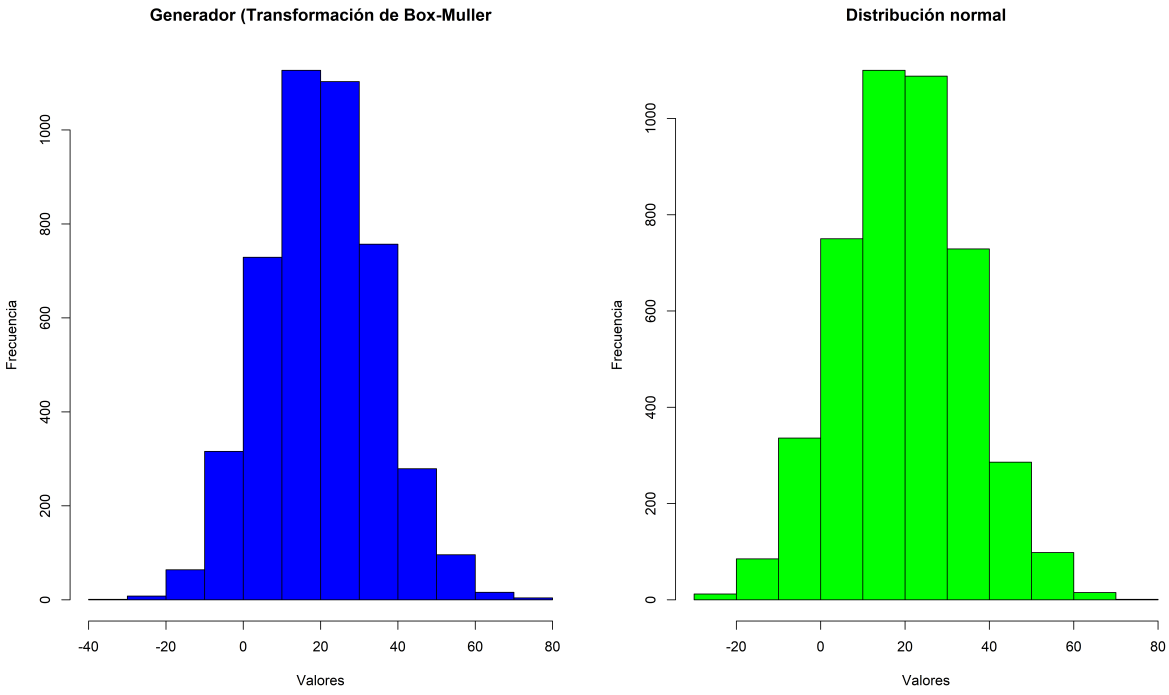

Figura 2: Comparativo de histogramas utilizando el generador propuesto (azul) y los resultados obtenidos por la función `rnorm` (verde), con $n = 4,500$ replicas, media ($\mu = 20$) y desviación estándar ($\sigma = 15$)

directa (recalculando la variable U_1, siempre que esta sea menor a U_2). En la figura 3, se muestran los histogramas de ambos casos comparados con los números aleatrios generados con la función `rnorm`. En la cual podemos observar que los histogramas cuando hay dependencia directa o indirecta entre las variables aleatorias uniformes U_1 y U_2 no son similares al histograma de los números aleatorios generados por la función función `rnorm`, lo que podría indicar que cuando hay dependencia entre U_1 y U_2, el generador propuesto, al parecer, no produce números aleatorios distribuidos normalmente. Este comportamiento se puede deber a que el generador (transformación de Box-Muller) funciona bajo el supuesto de que las variables U_1 y U_2 son números aleatorios uniformemente distribuidos, y una caracteristica de estos números es que deben ser estadísticamente independientes.

Para hacer el análisis de los resultados de los escenarios 2 y 3, se utilizó un diagrama de cajas y bigotes incluyendo también los resultados del generador considerando las variables z_0 y z_1 y la distribución normal con la función `rnorm`. Los resultados de esta experimentación se muestran en la figura 4. En la cual podemos observar que al ejecutar el generador sólo con la variables z_0, o con la variables z_1 o con ambas, no hay cambios significativos en los resultados obtenidos, por lo tanto, al parecer hay independencia entre estas variables, es decir, el generador puede ejecutarse solo con el cálculo de una de estas.

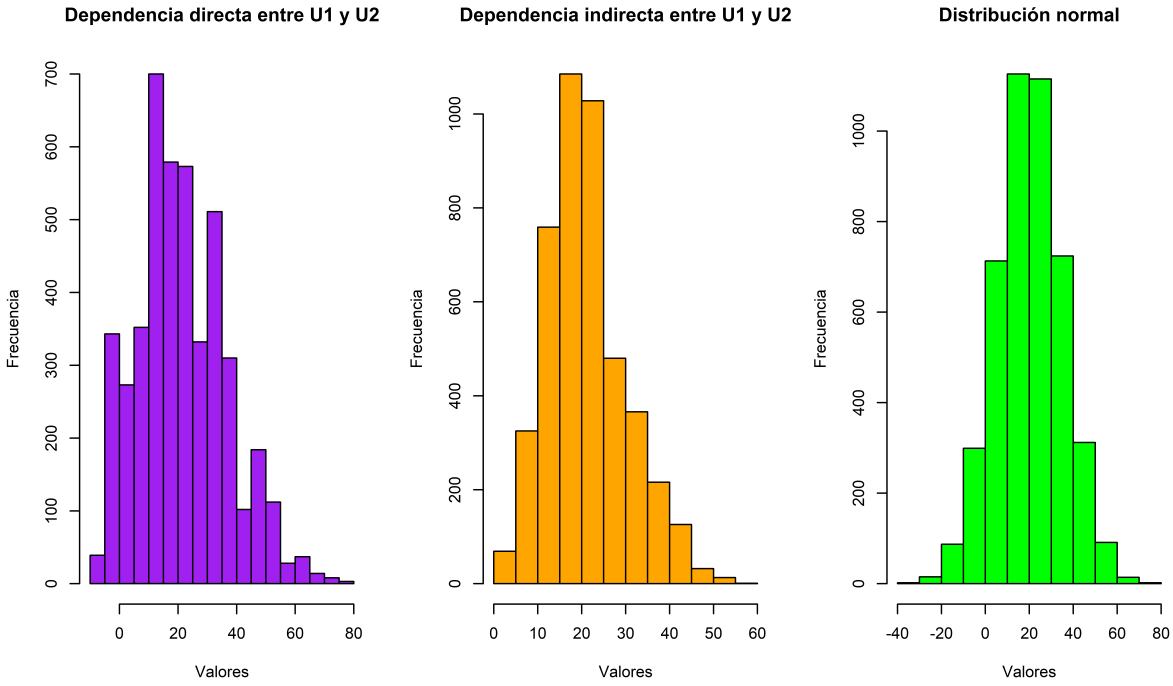

Figura 3: Comparación histogramas cuando en el generador las variables U_1 y U_2 tiene dependencia directa (morado), U_1 y U_2 tiene dependencia indirecta (naranja) y y los resultados obtenidos por la función `rnorm` (verde), con $n = 4,500$ replicas, media ($\mu = 20$) y desviación estándar ($\sigma = 15$)

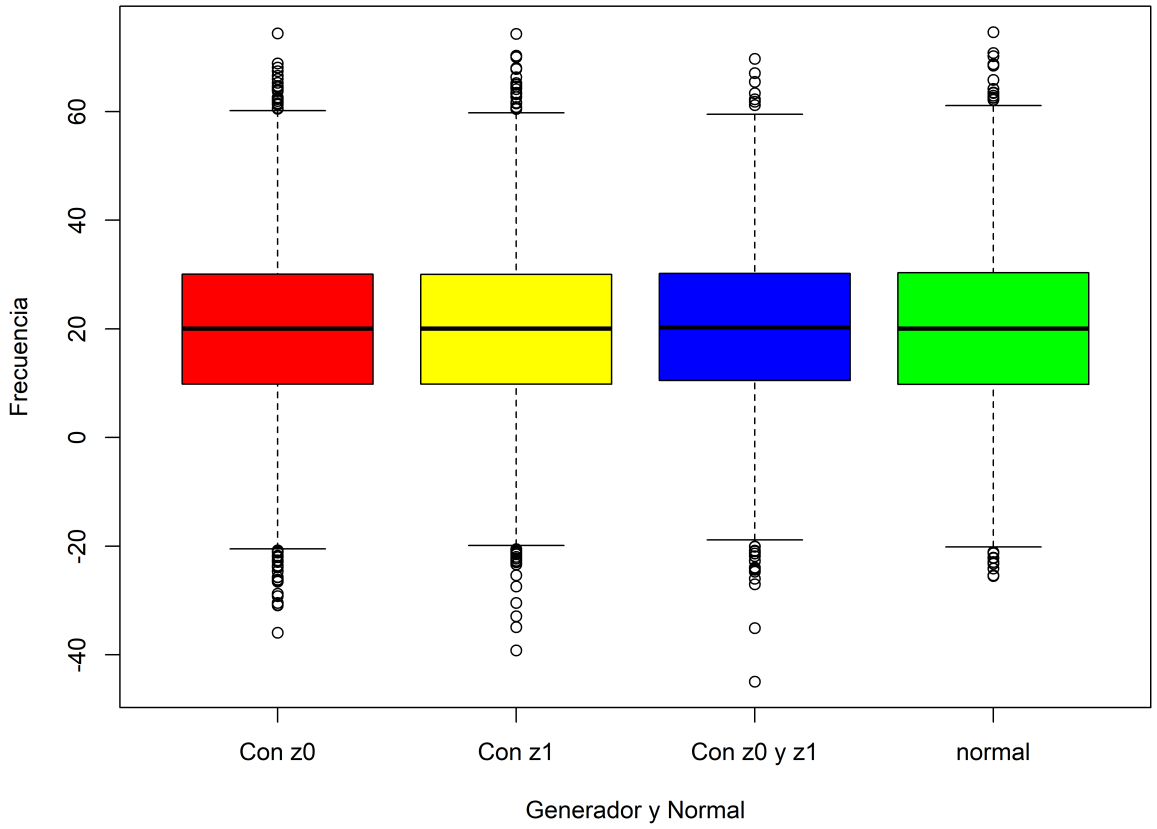

Figura 4: Comparación de diagramas de caja y bigotes del generador considerando sólo z_0 (rojo), con sólo z_1 (amarillo), ambas variables z_0 y z_1 (azul) y los resultados obtenidos por la función `rnorm` (verde), con $n = 4,500$ replicas, media ($\mu = 20$) y desviación estándar ($\sigma = 15$)

3.2. Uso del generador de números pseudoaleatorios con distribución uniforme en el generador de números pseudoaleatorios con distribución normal

A manera de práctica, se utiliza el generador de números pseudoaleatorios con distribución normal, modificando el algoritmo 1, de tal manera que las variables U_1 y U_2 se obtienen a partir del generador de números pseudoaleatorios con distribución uniforme tratado en la sección 2.

Para llevar a cabo la experimentación, se consideran los siguientes parámetros para el generador de distribución uniforme: $X_0 = 21$, $a = 8,001$, $c = 2,651$ y $m = 2,048$ con $n = 2$ réplicas y para el generador para los de distribución normal, $n = 4,500$ replicas, media ($\mu = 20$) y desviación estándar ($\sigma = 15$). Los resultados obtenidos de esta experimentación son mostrados en la figura 5. En la que podemos observar que los datos obtenidos con el generador se ajustan a una la distribución normal y, por lo tanto, el generador de números pseudoaleatorios con distribución uniforme, está arrojando valores con está distribución.

Referencias

[1] Bolaños Z., Johanna. Repositorio en GitHub de la clase de modelos probabilistas aplicados. Recursos libre, disponible en github.com/JohannaBZ/Probabilidad/tree/master/Tarea5, 2020.

[2] Box, G. E. P., Muller, Mervin E. A Note on the Generation of Random Normal Deviates. *The Annals of Mathematical Statistics*, 29(2), 1958.

[3] Coss, Raúl Bu. *Simulación. Un enfoque práctico.* EDITORIAL LIMUSA, S.A. de C.V., 2003.

[4] García D., Eduardo, García R., Heriberto, Cárdenas B., Leopoldo E. *Simulación y análsis de sistemas con Promodel.* Pearson Educación, 2006.

[5] Hillier, Frederick S., Lieberman, Gerald J. *Introducción a la investigación de operaciones.* McGraw-Hill, 9th edition, 2010.

[6] Schaeffer, Elisa. Modelos probabilistas aplicados: notas del curso. Recurso disponible en, https://elisa.dyndns-web.com/teaching/prob/pisis/prob.htmlut2.

[7] The R Foundation. The R Project for Statistical Computing. https://www.r-project.org/, 2020.

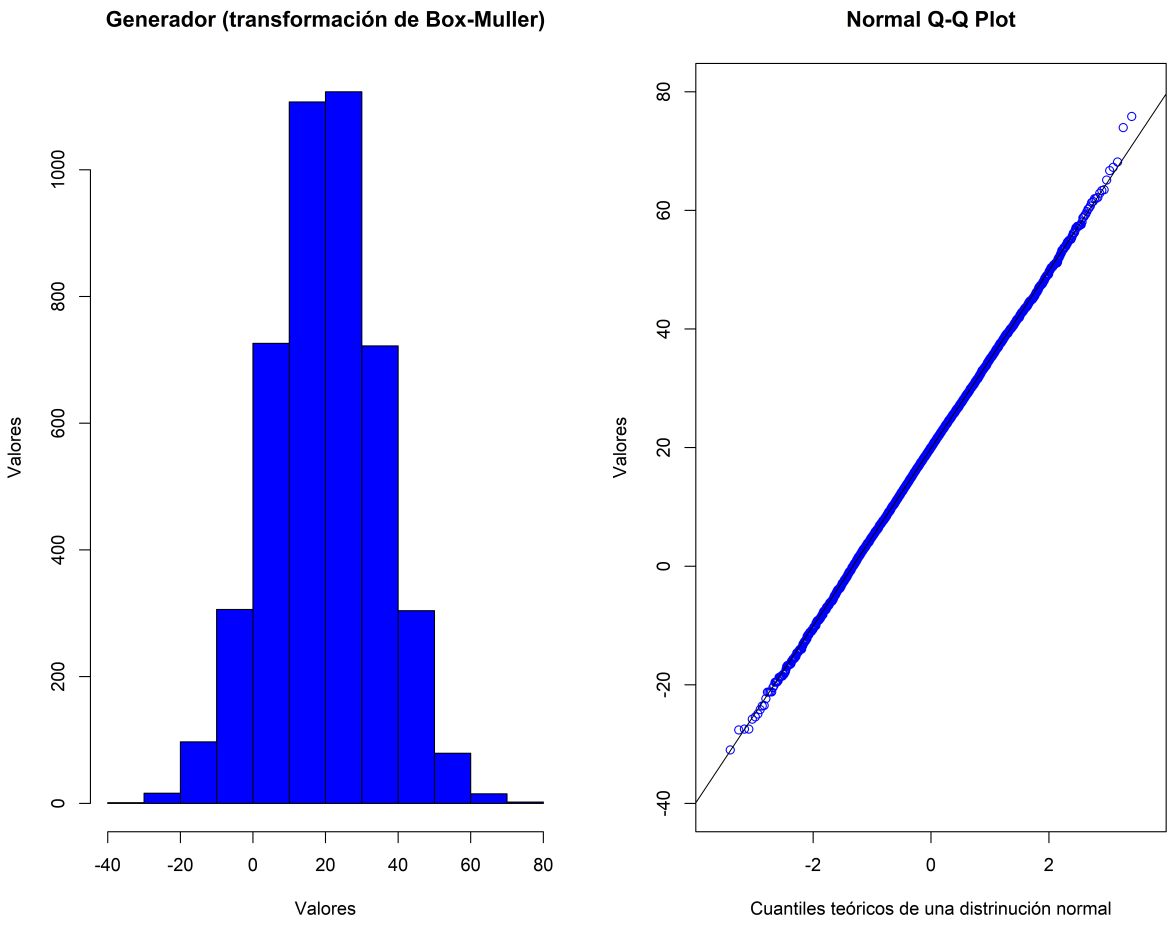

Figura 5: Correlación entre los valores obtenidos del generador (transformación de Box-Muller) y la distribución normal, con $n = 4,500$ replicas, media ($\mu = 20$) y desviación estándar ($\sigma = 15$)